i.A. Prof. Dr. Roland Füss SS 2008

Statistik II

Kurze Einführung in *TSP*

1. Allgemeines

Wir haben beim Arbeiten mit dem *TSP* zwei Möglichkeiten: Zum einen können wir *interaktiv* arbeiten, d.h. wir geben einen Befehl in eine *TSP*-Befehlszeile ein und lassen ihn sofort ausführen. Alternativ kann man ein Programm schreiben, das dann als ganzes ausgeführt wird (sogenannter *Batch-Modus*). Wir arbeiten wie in der Wissenschaft üblich mit Programmen (vgl. Abschnitt 3.1), um unsere Ergebnisse jederzeit reproduzieren zu können.

Das Rechenprogramm *TSP* läuft grundsätzlich mit 2 alternativen Benutzeroberflächen:

1. Das Aufrufen von und Arbeiten mit *„TSP through the Looking Glass“* (*TLG*) hat den Vorteil, Programminput und -Output parallel bearbeiten zu können. Leider kann *TSP* in diesem Modus nur sehr rudimentäre Graphiken als Vollbildschirm erstellen.

2. Bei Gebrauch der *GiveWin*-Oberfläche erscheinen Input und Outputs jeweils in anderen Fenstern. Dafür bietet diese Alternative volle Graphik-Funktionalität.

Wenn Sie Fragen bezüglich irgendwelcher Befehle in *TSP* haben, lohnt es sich immer, zunächst die umfangreiche Hilfefunktion (Menü: **Help**) zu verwenden oder aber in den *User's Guide* unter http://www.tspintl.com/products/ug_online.htm bzw. in das *Reference Manual* unter http://www.tspintl.com/products/rm_online.htm zu schauen.

2. Programme

TSP-Programme sind Textdateien, die mehrere *TSP*-Befehle enthalten und die Endung **.tsp** haben. Das Arbeiten mit Programmen hat den großen Vorteil, dass man die Ergebnisse später replizieren oder die Analyse ohne großen Aufwand abändern kann. Das Schreiben von

Programmen ist eine Kunst, die man nur durch Übung erlernen kann. Ein wichtiges Kriterium für die Güte eines Programms ist, dass es übersichtlich und gut dokumentiert ist. Deshalb sollte man nicht mit Leerzeilen und Tabs sparen und alle Schritte mit Kommentaren versehen. Kommentare sind Programmteile, die *TSP* nicht als Befehle interpretiert, sondern einfach ignoriert. Sie werden am Zeilenbeginn durch ein Fragezeichen eingeleitet.

Beispiel: `?Dies ist ein Kommentar`

Generierung eines Programms: Ein Programm kann in einem beliebigen Texteditor erstellt werden. Es muss als Textdatei mit der Extension **.tsp** abgespeichert werden (dazu im Dialogfeld „Speichern“ die Endung .tsp an den gewünschten Dateinamen anhängen und das ganze in „“ setzen). Alternativ kann man direkt in TSP Programme erstellen, und zwar unter *TLG* im Menü: **New Input** bzw. unter *GiveWin*: **File – New TSP Programm**.

Öffnen eines Programms: Ein existierendes Programm öffnet man entweder mit TSP über **File – Open** (*TLG*) oder **File – Open Text File** (*GiveWin*) bzw. über das entsprechende Symbol oder indem man im Windows-Explorer mit der rechten Maustaste auf die Datei klickt und mit „Öffnen mit“ die Anwendung, mit der das Programm geöffnet werden soll auswählt (z.B. einen Texteditor).

Starten eines Programms: Unter *TLG*: Klick auf das TSP-Symbol, unter *GiveWin*: Run-Button (laufendes Männchen).

TSP erstellt automatisch Outputfiles, die unter demselben Dateinamen mit der Endung **.out** abgespeichert werden.

Beenden eines Programms: Der Befehl `end;` beendet sowohl den *TSP* Programmabschnitt als auch den *TSP* Datenabschnitt.

Aufbau eines Programms:

```
options crt;

?Laden der Daten:
freq A; ?Jährliche Häufigkeit
smpl 1960 1998; ?Stichprobe über die Jahre 1960 bis 1998

read (FILE='vgrsekt.xls'); ?Lese eine Excel-Datei ein

?Speichern der im folgenden generierten Variablen:
out vgrsekt; ?Die Daten ausschreiben in eine TSP-Datenbank "vrgsekt.tlb"



?Ansehen der Daten
print  BAN1   BAN2   BAN3   BAN4   BAN5   BAN6   BAN7   BAN8   BAN9   BAN10
       WSN1   WSN2   WSN3   WSN4   WSN5   WSN6   WSN7   WSN8   WSN9   WSN10
       WSR1   WSR2   WSR3   WSR4   WSR5   WSR6   WSR7   WSR8   WSR9
WSR10;

?Generierung einer neuen Variablen:
?Beispiel: Berechnung der gesamtwirtschaftlichen Wertschöpfung

genr WSNGES = WSN1+WSN2+WSN3+WSN4+WSN5+WSN6+WSN7+WSN8+WSN9+WSN10;

?Beschreibende Statistik
msd WSNGES;

?Festlegung der Arbeitsstichprobe:
smpl 1970 1993;

?Graphische Darstellung der Daten:
plot(TITLE='Entwicklung de nominalen BIP in West-Dtld. seit 1970') WSNGES;

end;
```

3. Datensätze

TSP kann Datensätze in allen möglichen Formaten einlesen. Wir werden entweder Datensätze verwenden, die im Textformat (***.txt**), im MS Excelformat (***.xls**) oder im *TSP*-Format (***.tlb**) gespeichert sind. Datensätze, die in *TSP*-fremden Formaten vorliegen, werden mit `read` oder `load` eingelesen. *TSP*-Datensätze (interne Datenbank) hingegen mit `in`.

Beispiele: `read (FILE='vgrsekt.xls');` liest vgrsekt.xls ein
`in umfrage;` Liest umfrage.tlb ein

Den Befehl `out` kann man verwenden, wenn man den Datensatz einschließlich der neu generierten Variablen im *TSP*-Format (als TSP interne Datenbank mit Endung ***.tlb**) abspeichern will.

Vor dem Einlesebefehl sollten allerdings eine Reihe von anderen Befehlen stehen:

1) Der Befehl `options crt;` dient zur Formatierung des Outputs.

2) Die Frequenz des Datensatzes sollte vorgegeben werden, um die Stichprobe zu bezeichnen.

Mit dem Befehl `freq` legt man diese Frequenz fest.

Beispiel: `freq a;` jährliche Daten

3) Mit dem Befehl `smpl` legt man die Stichprobe fest.

Beispiel: `smpl 1960 1998;`

Der `smpl`-Befehl kann auch mit einer Bedingung verknüpft werden. Hierzu verwendet man den Befehl `smplif` oder den Befehl `select`. Der Hauptunterschied zwischen diesen beiden Befehlen ist, dass smplif kumulativ wirkt, während `select` immer wieder von der durch `smpl` oder `smplif` eingestellten Arbeitsstichprobe ausgeht. Um das Sample wieder auf die Gesamtstichprobe zurückzustellen, muss man nach `smplif` den `smpl`-Befehl verwenden, bei dem `select`-Befehl schreibt man hingegen einfach: `select 1;`

Beispiel: (Die beiden Seiten liefern äquivalente Ergebnisse!)

```
?Arbeitsstichprobe
smpl 1 200;

?Deskriptive Statistiken fuer
?Teilstichproben
smplif X>0;
   msd Y;

smplif X<=10;
   msd Y;

?Zurueckstellen der Arbeitsstichprobe
smpl 1 200;
```

```
?Arbeitsstichprobe
smpl 1 200;

?Deskriptive Statistiken fuer
?Teilstichproben
select X>0;
   msd Y;

select X>0 & X<=10; ?logisches Und
   msd Y;

?Zurueckstellen der Arbeitsstichprobe
select 1;
```

Im allgemeinen ist der `select`-Befehl dem `smplif`-Befehl vorzuziehen, da es in langen Programmen mitunter sehr schwer sein kann nachzuvollziehen, in welcher Weise man die Stichprobe durch eine Reihe aufeinanderfolgender `smplif`-Befehle eingeschränkt hat.

Hinweis: Eine der häufigsten Fehlerquellen ist, dass man die falsche Arbeitsstichprobe verwendet. Es empfiehlt sich daher stets zu überprüfen, ob Operationen auf der richtigen Stichprobe basieren.

Will man sich die in einem Datensatz enthaltenen Datenreihen ansehen, so verwendet man den `print`-Befehl.

Hinweis: Wenn Sie neue Variablen erzeugen, werden diese nur für die in dem Moment eingestellte Stichprobe erstellt!

4. Allgemeine Syntax

Im Folgenden zählen wir einige allgemeine Syntaxregeln auf, die beim Programmieren in *TSP* wichtig sind:

1. *TSP* unterscheidet nicht zwischen Groß- und Kleinbuchstaben.

   Beispiel: **CON**, **con** und **cOn** sind für *TSP* dieselbe Variable.

2. Jeder Befehl muss mit einem Semikolon abgeschlossen werden. (Dies ist eine sehr häufige Fehlerquelle!)

3. Befehle können generell abgekürzt werden, wenn die Abkürzung eindeutig ist. Wenn man sich nicht sicher ist, sollte man lieber den vollständigen Befehl hinschreiben.

   Beispiel: `ols` statt `olsq`

   Hinweis: Der Befehl `genr` zur Generierung von Variablen kann ganz weggelassen werden!

4. Wenn nichts anderes angegeben wird, geht *TSP* davon aus, dass neu generierte Variablen dieselbe Dimension haben sollen wie die anderen Variablen im Datensatz. Will man hingegen einen Skalar (Dimension 1×1) erzeugen, so muss man hierzu den Befehl `set` verwenden (häufige Fehlerquelle!).

Beispiel:	`msd X;`	berechnet die deskriptiven Statistiken der Variable **X** und speichert sie ab
	`set MEANX1 = @mean;`	erzeugt einen Skalar mit dem Mittelwert von **X**
	`MEANX2 = @mean;`	erzeugt hingegen eine Reihe mit derselben Dimension wie **X**, die überall den Mittelwert von **X** enthält

Hinweis: Man darf einen Variablennamen nur einmal vergeben, auch wenn die eine Variable ein Skalar und die andere eine Reihe ist.

5. Erzeugung neuer Datenreihen/Variablen

Neue Variablen lassen sich in *TSP* sehr einfach erzeugen, indem man einfach die Formel der neuen Variable aufschreibt.

Beispiel: `LOGWAGE = log(WAGE);`

Dieser Befehl erzeugt eine Variable **LOGWAGE**, die dem natürlichen Logarithmus der Variablen **WAGE** entspricht. Existiert bereits eine Variable **LOGWAGE**, so wird diese durch den obigen Befehl ohne Warnung überschrieben!

Die generierte Variable hat dieselbe Dimension wie die anderen Variablen des Datensatzes. Ist beispielsweise die Variable **WAGE** eine Zeitreihe über 100 Monate, so gilt das gleiche für die Variable **LOGWAGE**. Beachten Sie, dass die neue Variable nur für das aktive Sample erzeugt wird.

Will man eine Variable löschen, so verwendet man den Befehl `delete`.

Einige wichtige Funktionen/Operatoren:

`log(varname)`	natürlicher Logarithmus
`exp(varname)`	Exponentialfunktion
`abs(varname)`	Absolutbetrag
`sqrt(varname)`	Wurzel
`+`	Addition
`-`	Subtraktion
`*`	Multiplikation
`/`	Division
`varname**2`	Quadrierung
`=` oder `.eq.`	gleich (Bedingung)
`>` oder `.gt.`	größer (Bedingung)

`< oder .lt.`	kleiner (Bedingung)
`^= oder .ne.`	ungleich (Bedingungen)
`>= oder .ge.`	größer gleich (Bedingung)
`<= oder .le.`	kleiner gleich (Bedingung)
`& oder .and.`	"Und"-Operator
`| oder .or.`	"Oder"-Operator
`^ oder .not.`	"Nicht"-Operator
`varname(-1)`	Lag
`varname(+1)`	Lead

Bei der Generierung neuer Variablen ist zu beachten, dass der Variablenname bestimmte Bedingungen zu erfüllen hat:

1. Er muss mit einem Buchstaben, _, #, % oder @ beginnen.

2. Nachfolgende Zeichen dürfen Buchstaben oder Zahlen sein.

3. Er darf maximal 64 Zeichen haben.

In *TSP* kann man auch sehr einfach *Dummyvariablen* generieren.

Beispiel: Der Befehl `Y = X>0;` generiert eine Variable, die immer dann gleich eins ist, wenn **X**>0, und sonst gleich null.

6. Deskriptive Statistiken

Die deskriptiven Statistiken einer Variable kann man sich über den Befehl `msd` anzeigen lassen. Gleichzeitig werden diese Statistiken unter den folgenden Namen abgespeichert:

`@mean`	Mittelwert
`@stddev`	Standardabweichung
`@min`	Minimum
`@max`	Maximum
`@sum`	Summe der Elemente
`@var`	Varianz
`@skew`	Schiefe
`@kurt`	Excess Kurtosis (Wölbung)

Der Befehl `msd(all)` liefert zusätzliche Statistiken, der Befehl `msd(terse)` nur die ersten vier.

Sollen deskriptive Statistiken für mehrere Variablen berechnet werden, ist es sinnvoll die Option `byvar` zu wählen.

Beispiel: `msd(byvar) varname1 varname2;`

Dann wird für jede Variable einzeln die deskriptiven Statistiken berechnet, unabhängig von der Datenverfügbarkeit bei den anderen Variablen. Auf die einzelnen Statistiken kann dann durch Indizierung zugegriffen werden: `@mean(2)` wäre der Mittelwert von **varname2**.

Hinweis: Wenn man auf eine dieser Statistiken zugreifen möchte, muss man immer zuerst den `msd`-Befehl durchführen!

7. Graphische Darstellung

Es gibt verschiedene Möglichkeiten, die Daten graphisch darzustellen. Der Befehl `plot` stellt die Entwicklung von Variablen über die Zeit bei Zeitreihendaten dar (i.a. gemäß der Reihenfolge der Daten – diese kann durch Sortierung mit dem Befehl `sort` verändert werden. Der Befehl `graph X Y;` zeichnet ein Diagramm mit den Werten von **X** auf der Abszisse und **Y** auf der Ordinate. Wenn man sinnvoll verknüpfte Kurven erhalten möchte, sollten die Daten nach **X** sortiert werden (Befehl: `sort(all) X Y;`). `hist` ermöglicht die Darstellung von Daten als Histogramm.

Hinweis: Wenn man Graphiken erstellen will, sollte man *TSP* immer über *GiveWin* laufen lassen.

8. IF, ELSE, THEN

Mit diesen Befehlen kann man Bedingungen programmieren. Die Syntax lautet typischerweise folgendermaßen:

```
if SCALAR > 10;
then;
      msd X1;
      ols Y c X1;
else;
      msd X2;
      ols Y c X2;
```

In der `if`-Abfrage muss ein Skalar stehen, eine Datenreihe ist hier unzulässig!

Der `if`-Befehl sollte möglichst nicht zur Programmierung von Schleifen verwendet werden. Hierzu sollte besser der `do`-Befehl verwendet werden.

Hinweis: Wenn Bedingungen auf komplette Reihen auferlegt werden sollen, dann sollten `smplif` oder `select` verwendet werden.

9. Programmieren von Schleifen

9.1. Die DO-Schleife

Mit dem `do`-Befehl kann eine Gruppe von TSP-Befehlen mehrfach ausgeführt werden. In der Standardform verwendet der `do`-Befehl eine Zählvariable, die sich bei jedem Durchgang um 1 erhöht und beim angegebenen Höchstwert aufhört: `do i=1 to 10;`

Der Befehl wird mit `enddo;` abgeschlossen.

9.2. Die DOT-Schleife

Diese symbolische Schleife wird verwendet, wenn man dieselben Befehle für mehrere Variablen durchführen will. Nehmen wir z.B. an, wir wollen Wachstumsraten für 10 Länder berechnen, die von 1 bis 10 nummeriert sind. Dies können wir zum einen für jedes Land einzeln machen:

```
?Berechnung von Wachstumsraten für 10 Laender

WR1 = (GDP1-GDP1(-1))/GDP1(-1);
WR2 = (GDP2-GDP2(-1))/GDP2(-1);
WR3 = (GDP3-GDP3(-1))/GDP3(-1);
WR4 = (GDP4-GDP4(-1))/GDP4(-1);
WR5 = (GDP5-GDP5(-1))/GDP5(-1);
WR6 = (GDP6-GDP6(-1))/GDP6(-1);
WR7 = (GDP7-GDP7(-1))/GDP7(-1);
WR8 = (GDP8-GDP8(-1))/GDP8(-1);
WR9 = (GDP9-GDP9(-1))/GDP9(-1);
WR10 = (GDP10-GDP10(-1))/GDP10(-1);
```

Die `dot`-Schleife verkürzt die Programmierung enorm:

```
?Berechnung von Wachstumsraten für 10 Laender

dot 1-10;
    WR. = (GDP.-GDP.(-1))/GDP.(-1);
enddot;
```

In dem `dot`-Befehl steht immer eine Liste von Variablennamen oder von Teilen von Variablennamen. In den Befehlen, die für alle betroffenen Variablen durchgeführt werden sollen, werden diese Variablennamen oder Teile von Variablennamen durch einen Punkt ersetzt. Am Ende des Befehl muss ein `enddot;` stehen.

10. Matrixfunktionen in *TSP*

Mit dem Befehl `mmake` können aus einzelnen Variablen oder Reihen Matrizen aufgebaut werden. Wenn die Matrix auf Basis vorhandener Variablen erzeugt wird, dann entspricht die Anzahl von Zeilen der Matrix genau der Anzahl von Beobachtungen, die gerade mit `smpl` oder `select` eingestellt sind. Die Zahl der Spalten der Matrix entspricht genau der Anzahl von Variablen, die zu einer Matrix zusammengefasst werden. Soll eine Matrix aufgelöst werden, können die einzelnen Spalten mit dem Befehl `unmake` wieder zu Variablen oder Reihen umgeformt werden.

Beispiel:

```
?Aufbau einer Matrix mit mmake

?smpl legt die Länge der Reihen fest!
smpl 1 4;

?Einlesen von Datenreihen mit vier Werten
load X1;
1
2
3
4;
load X2;
9
8
7
6;

?Erzeugung einer 4x2-Matrix X aus den beiden Reihen
mmake X X1 X2;

?Unmake macht aus der Matrix wieder Reihen!

unmake X Z1 Z2;
```

Beim Befehl `unmake` muss festgelegt werden, welche Namen die Reihen/Variablen erhalten sollen, die aus den Spalten der Matrix erzeugt werden. Daher müssen hinter der aufzulösenden Matrix immer Variablennamen entsprechend der Spaltenanzahl folgen, in die die Spalten eingefügt werden.

Matrixoperationen lassen sich in *TSP* gut bewerkstelligen. Im Einzelnen gelten folgende Befehle für jede Matrix **m** (vor den Ausdrücken muss immer der Befehl `mat` verwendet werden, z.B. `mat m = m*m;`).

`m = m * m;` Matrixprodukt

`m = m * s;`	Skalarmultiplikation
`m = m';`	Transponierte einer Matrix
`m = m'';`	Inverse einer Matrix
`m = m # m;`	Kronecker-Produkt (⊗)
`m = m % m;`	Hadamard-Produkt (Element mal Element)

Skalare **s** aus Matrixoperationen sind ebenso einfach zu erzeugen: Auch hier muss zu Beginn der Befehl `mat` stehen, ein Skalar wird als 1×1-Matrix berechnet.

`s = DET(m);`	Determinante der Matrix
`s = TR(m);`	Spur der Matrix
`s = MIN(m);`	Element mit minimalem Wert
`s = MAX(m);`	Element mit maximalem Wert
`s = SUM(m);`	Summe der Elemente
`s = NROW(m);`	Anzahl der Zeilen
`s = NCOL(m);`	Anzahl der Spalten
`s = RANK(m);`	Rang der Matrix

Und schließlich noch das Beispiel für die Regressionsanalyse in Matrixschreibweise. Man erhält Regressionskoeffizienten über `mat b = (X'X)'' X' Y;`

11. Korrelation und Regression

Die Analyse von Zusammenhängen befasst sich mit rechnerisch feststellbaren, gleichzeitigen systematischen Veränderungen zweier Variablen. Hierbei sind Korrelation und Regression zentral.

11.1. Korrelation

Die Kovarianz ergibt sich gleichermaßen aus der Stärke eines Zusammenhangs zwischen zwei Variablen wie aus der Größe der Streuungen der einzelnen Variablen. Dabei weist das Vorzeichen auf die Richtung des Zusammenhangs hin: Ein positives Vorzeichen weist auf einen gleichlaufenden, ein negatives Vorzeichen auf einen gegenläufigen Zusammenhang zwischen zwei Variablen hin.

Um die Kovarianz als Zusammenhangskriterium nutzen zu können, ist der Einfluss der Varianzen von **X** und **Y** auf die Kovarianz herauszurechnen. Dies erreicht man, in dem die Kovarianz durch die Standardabweichungen dividiert wird. Man erhält damit den Korrelationskoeffizienten nach Bravais-Pearson, der per Definition zwischen -1 und +1 liegt (der Betrag der Kovarianz kann niemals das Produkt der Standardabweichungen überschreiten).

Mit dem Befehl `msd(terse,cova,corr) X Y;` erhält man nicht nur Mittelwert, Standardabweichung, Minimum und Maximum von beiden Variablen **X** und **Y**, sondern auch die Kovarianzmatrix (`@cova`) und die Korrelationsmatrix (`@corr`):

```
                        Results of Covariance procedure
                        ===============================

Number of Observations: 21
                   Mean       Std Dev       Minimum       Maximum
X              0.021300      0.072823      -0.11719       0.13143
Y               0.10110      0.027209      0.048493       0.15912

                                Covariance Matrix
                   X            Y
X            0.0053033
Y           -0.00023152    0.00074034

                               Correlation Matrix
                   X            Y
X                 1.0000
Y                -0.11684       1.00000
```

Um auf die berechneten Statistiken zugreifen zu können, müssen Indices eingesetzt werden:

Beispiel: `set a= @corr(1,2);`

Man hält mit **a** den Korrelationskoeffizienten zwischen der ersten und der zweiten Variablen, -0.11684, fest.

Hinweis: Die Kovarianz-Matrix und die Korrelation-Matrix sind symmetrische Matrizen. Die Elemente oberhalb der Hauptdiagonalen werden deshalb von *TSP* nicht extra ausgewiesen.

11.2. Regression

Der zentrale Befehl für die lineare Regression lautet olsq (*ordinary least squares*).

Beispiel: `olsq Y C X;`

Mit diesem Befehl regressiert man die abhängige Variable **Y** auf eine Konstante **C** und einen weiteren Regressor **X**. Als Output erhält man eine Tabelle mit den wichtigsten Regressionsergebnissen.

Beachten Sie, dass der Variablenname **C** reserviert ist für die Konstante. Diesen Namen darf man nicht selbst vergeben.

```
                                    Equation   1
                                    ============

                        Method of estimation = Ordinary Least Squares

Dependent variable: Y
Current sample:  1960:1 to 1994:4
```

```
Number of observations:  140

       Mean of dep. var. = 246.837      LM het. test = 12.6738 [.000]
  Std. dev. of dep. var. = 75.3640     Durbin-Watson = 1.67293 [.021,.031]
Sum of squared residuals = 5683.61  Jarque-Bera test = 3.89506 [.143]
   Variance of residuals = 41.1856   Ramsey's RESET2 = .634372 [.427]
Std. error of regression = 6.41760   F (zero slopes) = 19030.9 [.000]
              R-squared = .992801    Schwarz B.I.C. = 462.852
      Adjusted R-squared = .992749    Log likelihood = -457.910

           Estimated    Standard
Variable  Coefficient     Error       t-statistic   P-value
C         4.37769       1.83934       2.38003       [.019]
X         .859478       .623024E-02   137.953       [.000]
```

Nach jeder Regression werden die meisten der im Regressionsoutput angegebenen Statistiken und einige andere Variablen oder Vektoren abgespeichert. Die Variablennamen fangen üblicherweise mit @ an. Eine komplette Liste finden Sie im *Reference Manual*. Besonders wichtige abgespeicherte Größen Variablen sind:

`@coef`	geschätzter Koeffizientenvektor
`@res`	Reihe der Residuen
`@fit`	Reihe der gefitteten Werte